E1:246 Natural Language Understanding - Assignment 1
Word2Vec Skip gram model

Rahul Bansal
M.tech Ist year
rahulbansal@iisc.ac.in

Abstract

Vector space word representations are useful for many natural language processing applications. Data-driven learning of vector-space word embeddings that capture lexico-semantic properties is a technique of central importance in natural language processing.

Introduction

Word2vec is a two-layer neural net that processes text. Its input is a text corpus and its output is a set of vectors: feature vectors for words in that corpus. While Word2vec is not a deep neural network, it turns text into a numerical form that deep nets can understand. The purpose and usefulness of Word2vec is to group the vectors of similar words together in vectorspace. That is, it detects similarities mathematically. It does so in one of two ways, either using context to predict a target word (a method known as continuous bag of words, or CBOW), or using a word to predict a target context, which is called skip-gram. In the assignment, skip-gram model is implemented and explored.

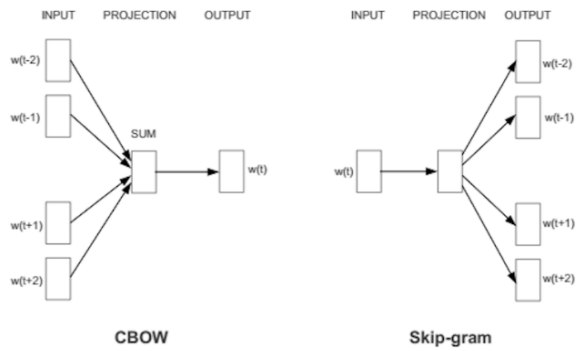

Figure 1

Skip-gram model

Mikolov et al. [1] introduced the Skip-gram model, an efficient method for learning high quality vector representations of words from large amounts of unstructured text data. The idea is to train the model for word-context pairs formed from text data. So ideally the words occuring in same context should have similar embeddings. Training skip gram model doesn't involve dense matrix multiplications as the model has only one hidden layer. The weights of hidden layer are actually word vectors we strive to learn.

Solution approach

The objective of the Skip-gram model is to find word representations that are useful for predicting the surrounding words in a sentence or a document. More formally, given a sequence of training words w1, w2, w3, . . . , wT , the objective of the Skip-gram model is to maximize the average log probability

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c}\log p\Big(\frac{w_{t+j}}{w_t}\Big)$$

where c is the size of the training context (which can be a function of the center word wt). Larger c results in more training examples and thus can lead to a higher accuracy, at the expense of the of training time. The basic Skip-gram formulation defines $p(w_{t+j}|w_t)$ using the softmax function:

$$p(\frac{w_O}{w_I}) = \frac{\exp {v'_{w_O}}^T v_{w_I}}{\sum_{w=1}^{W}\exp {v'_{w_O}}^T v_{w_I}}$$

where v_w and v'_w are the input and output vector representations of w, and W is the number of words in the vocabulary. This formulation is impractical because the cost of computing $\nabla \log p(w_O|w_I)$ is proportional to W which is often large(10^5 - 10^7terms). Negative sampling, hierarchical softmax is used to deal with this. [2]

Dataset

Training data is downloaded from reuters corpus[3]. Text processing is done on training data to remove noise in the data i.e punctuation words etc. Data pairs are generated by keeping a fixed window size.
Test data- ”questions-words.txt” for qualitative and quatitative reasoning is downloaded from the[4]. Test data contains word pairs exhibiting same kind of relationship.

Model architecture

To predict surrounding words in a sentence, a neural network with a single hidden layer is trained. The dimensions of the hidden layer is (vocab_size, embedding dims) and the dimesions of output layer is (embedding dims, vocab_size). The goal is actually just to learn the weights of the hidden layer, these weights are actually the word vectors that were trying to learn.There is no activation function on the hidden layer neurons, but the output layer use softmax.

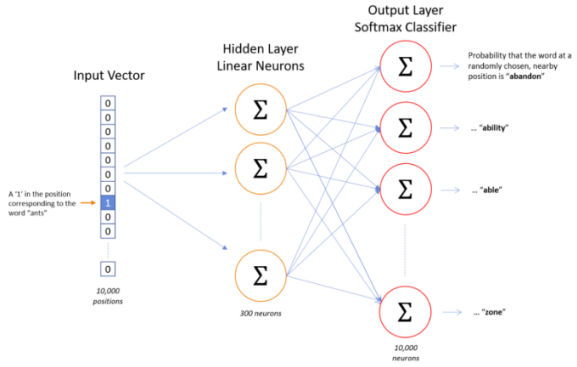

Figure 2: Model architecture
Courtesy: Medium.com

Negative sampling

As the size of our word vocabulary is very large, it means that our skip-gram neural network has a tremendous number of weights, all of which would be updated slightly by every one of our million of training samples.For this assigment, more than 4 million training samples are generated.
Negative sampling addresses this by having each training sample only modify a small percentage of the weights, rather than all of them. With negative sampling, small number of negative words are randomly selected (lets say 5) to update the weights for. (In this context, a negative word is one for which we want the network to output a 0 for). Weights for positive word still be updated

Steps for training

1. Raw training data is downloaded from reuters corpus. In raw data, there were occurences of words like ’.’ ’,’ ’single alphaet’ etc. keras.textpreprocessing is used to remove such word.
2. Vocaulary size is 26322.
3. Data pairs are generated with WINDOW_SIZE = 2.
4. word2int[] and int2word[] hash table are created so as to get one hot encoding for each word.
5. Model is trained for embedding_size = [64,128,256] and negative_samples = [2,3,4,5,6]. Embeddigs are saved for each of the combination of embedding_size and negative_sample.
6. tensorflow is used to make the model graph. Batch_size = 64 and models are trained for 50 epochs.

Observations

SimLex-999, a gold standard resource for evaluating distributional semantic models SimLex-999 contains a range of concrete and abstract adjective, noun and verb pairs, together with an independent rating of concreteness and (free) association strength for each pair. This diversity enables fine-grained analyses of the performance of models on concepts of different types, and consequently greater insight into how architectures can be improved.

Serial	Dataset	Embedding dims	Negative samples	Num pairs	Not found	Rho
1	Simlex	64	2	999	375	0.115799
2	Simlex	64	3	999	375	0.060817
3	Simlex	64	4	999	375	0.04563
4	Simlex	64	5	999	375	0.069535
5	Simlex	64	6	999	375	0.064626
6	Simlex	128	2	999	375	0.042036
7	Simlex	128	3	999	375	-0.00285
8	Simlex	128	4	999	375	0.115273
9	Simlex	128	5	999	375	0.047641
10	Simlex	128	6	999	375	0.099992
11	Simlex	256	2	999	375	0.041582
12	Simlex	256	3	999	375	0.050084
13	Simlex	256	4	999	375	0.00662
14	Simlex	256	5	999	375	0.019875

Figure 3

Figure 3 shows the spearman correlation for models trained for differet embedding_dims and number of negative samples.
As it is obeserved that maximum value of correlation happens to occur for embedding_size = 64 and negative_sample = 2.

Task 2

Qualitative reasoning

The word embeddings learnt from reuters corpus are tested on 'question-words.txt'. The TSNE plots are plotted by randomly considering 25, 50, 75 , 100 words from the 'question-words.txt'.

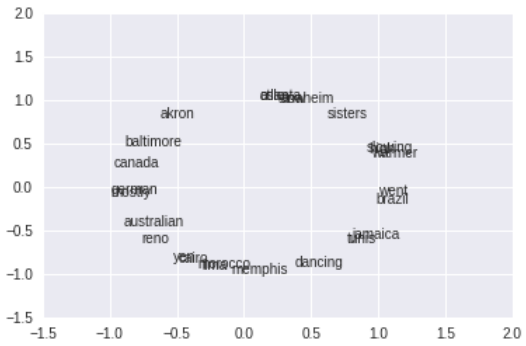

Figure 4: TSNE plot with 25 words

Figure 5: TSNE plot with 50 words

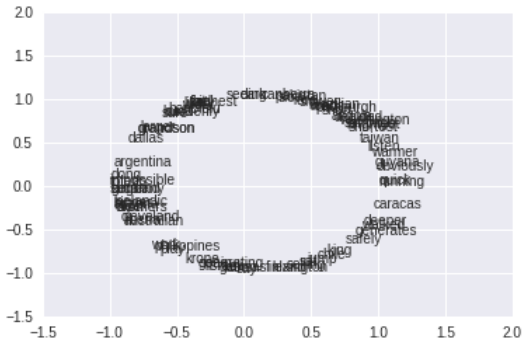

Figure 6: TSNE plot with 75 words

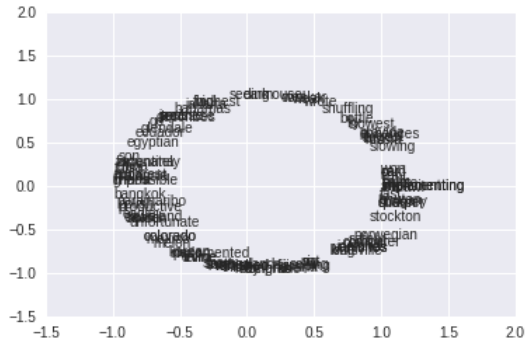

Figure 7: TSNE plot with 100words

Clusters of the words can be observed. Words which are closer to each other in 2-D space are semantically or syntactically similar.

Quantitative reasoning

As the maximum spearman correlation is observed when embedding dimension = 64 and number of negative samples = 2, so quantitative reasoning is done with word vectors generated with the same embedding dimensions and negative samples. Quantitative reasoning is done on "questions-words.txt".

There are four colums in text data- col1[i]:col2[i]::col3[i]:col4[i]. So what has to be done is to verify if col4[i] is present in the neighborhood of col2[i]-col1[i]+col3[i] or not. By increasing the length of neighborhood the number of matches is expected to increase. Neighborhood means all the vectors which are close to col2[i]-col1[i]+col3[i] in terms of euclidean distance.

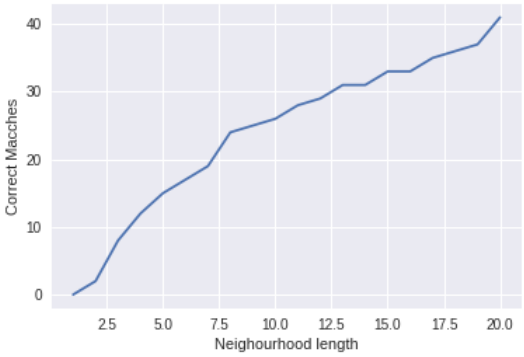

Figure 8

As we increase the length of the neighborhood, the number of correct matches increases.

Figure 9 shows the tabular form of correct matches vs length of neighborhood

Length of neighborhood	Correct Matches
1	0
2	2
3	8
4	12
5	15
6	17
7	19
8	24
9	25
10	26
11	28
12	29
13	31
14	31
15	33
16	33
17	35
18	36
19	37
20	41

Figure 9

Figure 10 shows the closest words in the vector space to some of the common words.

Word	Closest 5 words in vector space					
is	was	ccpt	anymore	unchangd	rabbit	
have	been	they	had	277p	perfect	
man	sumun	signing	metallverken	thdr	citizen	
child	justo	vlissingen	weapons	tse	caribou	
was	is	nonexclusive	cred	disruption	appreciations	
were	are	afternoons	resurgence	angelica	scrutiny	
he	slides	said	taipei	valve	unconditional	
she	speakes	sixteen	differing	ldmfa	reimpose	
king	vicorp	nold	unemotional	burger	converter	
queen	skanska	counterproductive	lsa	penobscot	interesdt	
satisfy	sovereignty	categorically	adjustment	tsunao	reassessment	
content	ack	512	substantiate	nisshin	incidents	
kind	invoked	surfaced	wd	nice	hmi	
stock	mmft	imtec	appreciating	igen	bbc	
news	briefing	agency	372p	categorically	cambist	
been	had	has	have	oshuf	prevents	
pretty	sixteen	avge	jeopardize	clearer	perished	

Figure 10

Figure 11 shows the closest words in the vector space to some of the country names.

Countries	Closest 5 words in vector space				
greece	compensation	bair	visits	come	ticking
india	lafayette	plaintiffs	illumination	fnhb	undue
france	recur	stna	knock	smoothed	viking
china	std	suna	countervailing	8750	boasting
germany	west	zuckerman	fog	tlci	anybody
mexico	distressing	hatakeyama	eventually	margoshes	nbty
pakistan	hugoton	cool	abnn	noil	obliged
japan	chesapeake	japanese	itr	sohio	kaufhof
korea	sunnier	harrowing	populous	napo	capability
brazil	bails	prov	promexpo	appearance	garj

Figure 11

References

[1] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space.

[2] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality.

[3] Reuters corpus: https://www.nltk.org/book/ch02.html.

[4] questions-words.txt: https://code.google.com/archive/p/word2vec/source/default/source.